Particle Filters for Single-objective Numerical Optimization

Milad Rostampour

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
August 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Zeki Bayram
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Adnan Acan
Co-Supervisor

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Mehmet Bodur

2. Assoc. Prof. Dr. Mehtap Köse Ulukök

3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

This thesis introduces a novel approach combining Particle Filters and the L-BFGS-B optimization method for solving single-objective numerical optimization problems. The proposed method intricately marries the stochastic exploration of Particle Filters with the local optimization prowess of L-BFGS-B to navigate complex landscapes efficiently. Extensive experimentation on benchmark problems validates the approach's effectiveness, convergence speed, accuracy, and robustness. This fusion of methodologies opens new vistas for conquering diverse optimization challenges.

Keywords: particle filters, evolutionary algorithms, optimization.

ÖZ

Bu tez, problemleri sürekli ve ayrık alanlar olarak sınıflandırmakta ve tek amaçlı sayısal eniyileme problemlerini çözmek için Parçacık Filtreleri ile L-BFGS-B eniyileme yöntemini birleştiren yeni bir yaklaşım sunmaktadır. Önerilen yöntem, karmaşık alanlarda verimli bir şekilde gezinmek için Parçacık Filtrelerinin stokastik keşfi ile L-BFGS-B'nin yerel eniyileme becerisini karmaşık bir şekilde birleştirmektedir. Karşılaştırmalı problemler üzerinde yapılan kapsamlı deneyler, yaklaşımın etkinliğini, yakınsama hızını, doğruluğunu ve sağlamlığını doğrulamaktadır. Metodolojilerin bu birleşimi, çeşitli optimizasyon zorluklarının üstesinden gelmek için yeni ufuklar açmaktadır.

Anahtar Kelimeler: parçacık filtreleri, evrimsel algoritmalar, eniyileme.

DEDICATION

In the pages of this thesis, a journey unfolds, a journey that could not have been ventured upon without the unwavering love, encouragement, and support of those who mean the world to me.

To my father, Morteza, whose boundless wisdom and strength have been my guiding light. You've taught me the power of determination and the beauty of pursuing knowledge. Your sacrifices and endless belief in me have shaped every word written here.

To my mother, Farkhondeh, whose grace and kindness infuse every aspect of my being. Your endless patience, unyielding faith, and gentle spirit have given me the courage to chase my dreams. Your unwavering love has given wings to my aspirations.

To my brother, Masih, and my sister, Mina, who have stood by my side through every challenge and triumph. Our shared laughter, our shared tears, they have been the foundation upon which I've built this academic endeavour. You've shown me the strength of familial bonds that nothing can break.

To my friends, Ehsan, Majid, and Afshin, whose camaraderie and support have been a constant source of inspiration. Through late nights of study and moments of doubt, your presence has reminded me that the path is best walked with kindred spirits.

ACKNOWLEDGMENT

I extend my deepest appreciation to my esteemed supervisors, Assist. Prof. Dr. Ahmet Ünveren and Assoc. Prof. Dr. Adnan Acan. Your exceptional mentorship, insightful feedback, and dedication to my research have not only fuelled my academic progress but have also honed my scholarly perspective.

To Prof. Dr. Mehdi Jabalameli from my bachelor's program, I owe a debt of gratitude for sparking my passion for academia. Your guidance and encouragement have set me on this path of continuous learning.

My heartfelt thanks go to the professors, research assistants, colleagues, and the entire computer engineering department who have contributed to my academic journey. Your collective expertise, camaraderie, and unwavering support have been invaluable in shaping my understanding of my field.

I am genuinely appreciative of the knowledge you have shared, the guidance you have offered, and the genuine interest you have shown in my academic pursuits. Your dedication to education and research has left an indelible mark on my academic journey, and I am honoured to have been a part of this scholarly community.

In closing, I am humbled by the pivotal role each individual has played in shaping my academic accomplishments. To all those who have shared their wisdom, time, and unwavering belief in my potential, I extend my deepest appreciation. Your impact will resonate in my academic pursuits for years to come.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

A fundamental human pursuit that has been around for generations is optimization. The search for the best answers has fuelled scientific research, technical development, and effective decision-making from the dawn of civilization to the current day. Finding the greatest options among alternatives has significant effects on a variety of fields, including engineering, economics, logistics, and more. Here, we examine the categorical differences among optimization issues, their historical development, and the nuances of single-objective numerical optimization problems [1].

1.1 Optimization Problems: A Historical and Categorical Overview

The origins of optimization can be found in prehistoric societies. The search for ideal areas and volumes shows that the ancient Greeks, for example, struggled with the optimization of geometric shapes. Early innovators like Archimedes used cutting-edge techniques to determine the best answers in real-world situations. But optimization didn't become a discipline with solid theoretical underpinnings and real-world applications until the development of formal mathematical modelling and computers.

The development of linear programming in the middle of the 20th century was a turning point that introduced optimization to the world of rigorous mathematical analysis. Linear programming enabled the optimization of linear objective functions subject to linear constraints, revolutionising fields such as resource allocation and transportation planning. The simplex method, developed by George Dantzig, played a

crucial role in solving linear programming problems and laid the groundwork for further optimization techniques [2, 3].

1.2 Categories of Optimization Problems

The nature of the choice variables, the existence of constraints, and the properties of the objective function are frequently used to describe the variety and complexity of optimization issues. Due to these traits, optimization issues are divided into two main groups: continuous optimization and discrete optimization.

1.2.1 Continuous Optimization Problems

Decision variables in continuous optimization problems can take on any real value within a given range. This category has a number of subcategories, each of which addresses particular optimization nuances:

1. Unconstrained Optimization: The objective of unconstrained optimization is to identify the optimal value of an objective function without imposing any restrictions on the variables used for making decisions. Gradient-based algorithms use derivatives of the objective function to direct the search for the optimum, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approach. The derivative-free method developed by Nelder-Mead iteratively updates a simplex to explore the solution space.

2. Constrained Optimization: Constrained optimization involves optimizing the objective function while adhering to a set of constraints. These constraints can be expressed as equality or inequality conditions on the decision variables. The Sequential Least Squares Quadratic Programming (SLSQP) method combines quadratic programming and least squares optimization to efficiently navigate constrained optimization problems. Trust region methods, such as trust-constr, construct a trust region around the current solution and iteratively refine it within the constraints [4, 5].

1.2.2 Discrete Optimization Problems

Discrete optimization problems revolve around decision variables that can only take on distinct values, often restricted to integers or binary choices. This category includes:

1. Integer Programming: In integer programming, decision variables are required to assume integer values. These problems find applications in scenarios like scheduling and resource allocation. The Travelling Salesman Problem (TSP), a classic example, seeks the shortest route visiting a set of cities and returning to the starting point.

2. Binary Optimization: Binary optimization involves decision variables restricted to binary values (0 or 1). The Knapsack Problem exemplifies this category, where a knapsack has limited capacity, and the goal is to select items to maximise the total value while staying within the capacity constraint [6, 7].

1.2.3 Comparison and Trade-offs

Continuous optimization methods leverage the smoothness of objective functions and constraints, often converging rapidly in continuous domains. However, they may struggle in cases involving discrete or integer decision variables, requiring techniques to handle these situations effectively.

Discrete optimization methods excel when dealing with inherently discrete decision variables. They are well-suited for combinatorial problems that demand solutions satisfying discrete conditions. Yet, these methods can encounter challenges when addressing complex continuous landscapes or nonlinear objective functions.

Real-world optimization problems often exhibit a mix of continuous and discrete decision variables. Hybrid approaches, such as mixed-integer programming and genetic algorithms, bridge the gap between these two domains, effectively handling mixed-type problems.

1.3 Particle Filters: A Bayesian Approach to Optimization

Particle Filters, also known as Sequential Monte Carlo methods, represent a powerful paradigm for addressing dynamic and uncertain environments. Initially developed for state estimation and tracking problems, Particle Filters have found versatile applications, including optimization. This section delves into the foundational concepts of Particle Filters, their components, and their role in optimization.

1.3.1 Particle Filters: Core Concepts and Components

At the heart of Particle Filters lies the representation of probability distributions through a set of particles. Each particle encapsulates a state hypothesis that collectively approximates the true underlying distribution. Particle Filters operate through distinct stages:

1. Initialization: The process begins by initialising particles with state hypotheses, often drawn randomly from an initial state distribution. This step establishes the initial exploration of the solution space.

2. Prediction: Particles are propagated through a dynamic model that characterises the system's evolution. The model accounts for system dynamics and noise, projecting particles forward in time.

3. Update: The inclusion of measurements is crucial to the effectiveness of particle filters. The weights of the particles are modified according to their likelihood in relation to the measured values. Higher weights are given to particles that are more likely.

4. Resampling: When resampling, particles with greater weights are more likely to be chosen for the following generation, resulting in more accurate and representative posterior distribution particles.

5. Estimation: Finally, weighted averages or other estimating methods based on the states and weights of the particles are used to estimate the state of the system. This estimation represents the solution to the optimization problem.

In the figure below, a simple flowchart of the mentioned steps is illustrated:

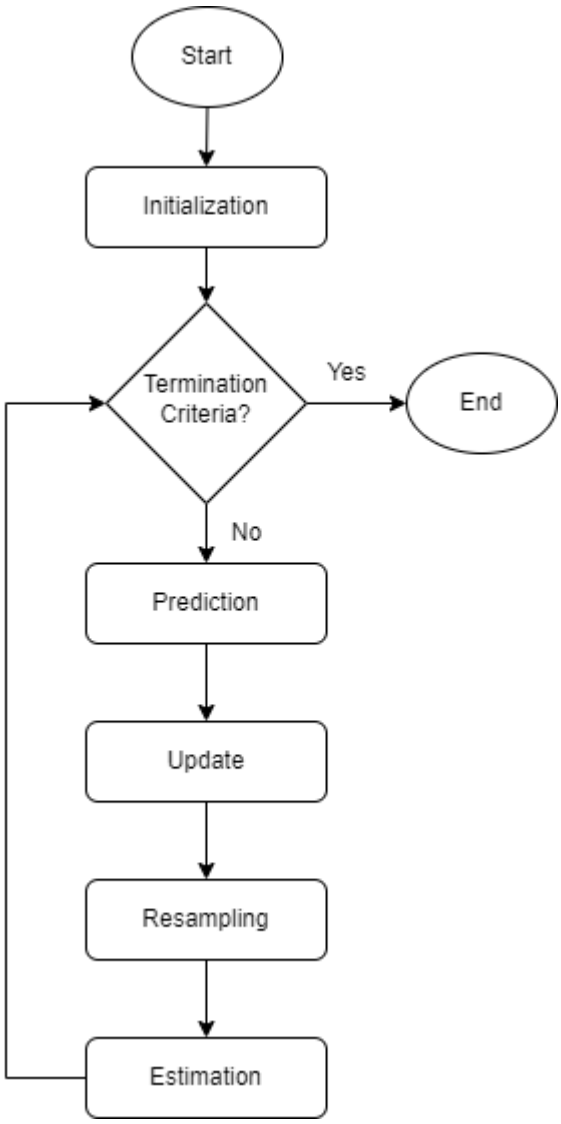

Figure 1.1: Particle Filters Flowchart

1.3.2 Particle Filters: A Bayesian Approach to Optimization

A key component of Bayesian filtering, particle filters have become an effective method for managing uncertainty and dynamics in a variety of applications, including optimization. In-depth discussion of the historical development of particle filters, their functions, and a comparison to earlier strategies like Kalman filters are provided in this section.

1.3.3 Particle Filters: Historical Evolution and Comparison

Particle filters have their origins in Nils Aall Barricelli's development of the Sequential Monte Carlo (SMC) approach in the 1950s. The technique was designed to mimic how interacting particles behave in biological systems. However, it wasn't until the 1990s that Particle Filters became well-known in the state estimation field, thanks to the computer vision research of Michael Isard and Andrew Blake.

Particle Filters provide a novel viewpoint in the optimization environment when compared to traditional methods like Kalman Filters. The state distribution is assumed to be gaussian by Kalman filters, which perform well in linear Gaussian systems. Particle Filters, on the other hand, embrace nonlinearity, multimodality, and non-Gaussian distributions. This adaptability makes Particle Filters an appealing choice for optimization problems with complex landscapes and uncertain dynamics.

1.3.4 Particle Filters: Functionalities and Examples

Particle Filters exhibit several key functionalities that make them a versatile tool for optimization:

1. Handling Nonlinearity: Particle Filters shine in scenarios where the objective function exhibits nonlinear behaviour. Their probabilistic approach allows particles to explore nonlinear solution spaces effectively.

2. Multimodal Solutions: Particle Filters naturally address multimodal landscapes by maintaining diverse particles that represent different modes of the distribution. This property is crucial in optimization problems with multiple solutions.

3. Uncertainty Incorporation: Particle Filters seamlessly integrate uncertainties in both the system dynamics and measurements. This is particularly valuable in optimization tasks where noisy data and uncertain parameters are involved.

4. Complex Dynamics: Particle Filters thrive in problems with intricate dynamics, as they capture the evolution of particles over time through prediction and update stages.

1.3.5 Particle Filters: Example Application

Consider an optimization scenario in robotics, where a mobile robot aims to navigate an environment and find the optimal path that minimises energy consumption while avoiding obstacles. Traditional optimization methods may struggle with the nonlinear and uncertain nature of the problem. Particle Filters offer an elegant solution by modelling the robot's motion as a dynamic system and using sensor measurements to update the particles representing possible paths. This approach not only optimises the path but also provides insights into uncertainty levels, aiding decision-making.

1.3.6 Particle Filters: The Optimization Advantage

Particle Filters introduce a probabilistic lens to optimization, allowing for the exploration of complex landscapes, incorporation of uncertainties, and handling of nonlinear and non-Gaussian distributions. Their adaptability presents them as a powerful alternative to conventional optimization techniques, particularly in circumstances where those techniques might fail.

1.4 Optimization Methods: A Comprehensive Overview

The many-faceted difficulties of optimization problems are addressed by a wide variety of optimization strategies in addition to Particle Filters. The methodologies, convergence characteristics, and appropriateness of these methods for various problem structures vary. An extensive examination of the main optimization techniques is provided in this section.

1.4.1 Nelder-Mead Method

The Nelder-Mead algorithm, sometimes referred to as the Downhill Simplex Method, is an unconstrained direct search optimization method. It functions without the need for derivative knowledge and iteratively manipulates a simplex, a geometric structure with vertices that designates a location in the search space, to explore the solution space. Based on the function evaluations at the simplex's vertices, the algorithm conducts reflection, expansion, contraction, or reduction operations on the simplex at each iteration. Nelder-Mead is best suited for functions with erratic topographies, discontinuities, or in the absence of derivative information [4].

1.4.2 BFGS Method: Approximating the Hessian

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) approach, which focuses on unconstrained problems, is a cornerstone of optimization. Gradients are used by BFGS to estimate the Hessian matrix, which is the second derivative of the objective function. For objective functions that behave well, this approximation provides rapid convergence during iterative optimization.

1.4.3 Powell's Method

Finding the local minima of unconstrained optimization problems is the goal of Powell's method, which is also known as the conjugate direction method. It works by combining past search directions to create conjugate directions, which are then

iteratively searched along. Powell's approach is particularly successful when the objective function is not smooth and can efficiently navigate through solution spaces that are extremely curved [5].

1.4.4 Conjugate Gradient (CG) Method

The Conjugate Gradient method is an iterative approach created for issues involving unrestricted optimization. It uses gradient information to iteratively locate the best answer, modifying search directions by making sure they are conjugate with previous directions. CG is frequently employed in large-scale optimization issues, including those that arise in machine learning and image processing, and is well-suited for quadratic objective functions [8].

1.4.5 Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method

A quasi-Newton approach for unconstrained optimization is the BFGS algorithm. It uses gradient information to roughly estimate the goal function's Hessian matrix and then iteratively improves this estimate. BFGS is a well-liked option for optimising smooth nonlinear functions because it combines the advantages of gradient-based techniques with the capacity to handle complicated topographies [9-11].

1.4.6 Newton-CG Method

In order to optimize unconstrained problems, the Newton-CG technique combines the Conjugate Gradient method and Newton's approach, which employs second derivatives. Within a trust region, it looks for the quadratic approximation of the objective function that is minimised. In comparison to gradient-based approaches, this method has faster convergence characteristics and can handle nonlinear objective functions [12].

1.4.7 Limited-memory BFGS with Bounds (L-BFGS-B) Method

A modification of the BFGS method designed for bound-constrained optimization issues is called L-BFGS-B. To conserve computational resources, it keeps a Hessian matrix approximation in restricted memory. Within variable boundaries, L-BFGS-B efficiently explores the solution space specially for problems with constraints [13].

1.4.8 Trust Region Reflective Newton-CG (TRON) Method

Large-scale unconstrained optimization issues can be resolved with the Trust Region Reflective Newton-CG approach. Within a trust region framework, it combines Newton's approach and the Conjugate Gradient method. TRON maintains a balance between exploration and exploitation by adjusting its strategy dependent on the local geometry of the objective function [14].

1.4.9 COBYLA Method: Derivative-Free Constrained Optimization

The Constrained Optimization BY Linear Approximations (COBYLA) method is designed for derivative-free constrained optimization problems. COBYLA iteratively constructs linear approximations to the objective function and constraints, exploring the solution space efficiently. It is suitable for scenarios where derivative information is challenging to compute [15].

1.4.10 Sequential Least Squares Quadratic Programming (SLSQP) Method

SLSQP is a flexible optimization technique that combines the concepts of least squares and quadratic programming for constrained problems. It seeks to minimise a quadratic approximation of the objective function while adhering to linear equality and inequality constraints. SLSQP can efficiently navigate complex landscapes with constraints [16].

1.4.11 Trust Region Constrained (trust-constr) Method

Trust Region Constrained is a versatile optimization method that handles nonlinearly constrained problems. It operates by iteratively approximating the objective function and constraints using quadratic models within trust regions. The method adapts its trust regions to ensure both global and local convergence [17].

Chapter 2

LITERATURE REVIEW

2.1 Optimization Approaches for Single Objective Numerical Problems

Optimization problems have attracted significant attention across a spectrum of fields due to their profound impact on decision-making and resource allocation. This literature review delves into various optimization methodologies employed for solving single objective numerical problems, shedding light on their mechanics and applications.

2.1.1 Classical Optimization Methods

Gradient-Based Methods: Gradient-based optimization methods leverage the gradient, or derivative, of the objective function to iteratively guide the search towards the optimum. The steepest descent method, for instance, moves in the direction of the negative gradient, following the path of steepest descent. Newton's method incorporates second-order information, such as the curvature of the function, to adjust the step size more intelligently. Conjugate gradient methods, on the other hand, maintain conjugacy among search directions, allowing for efficient exploration of the solution space.

Example of Steepest Descent Method: Consider a quadratic function **f(x) = x^2 - 6x + 5**. The gradient is **f'(x) = 2x - 6**. Starting at **x = 5**, the steepest descent direction is **x = 5 – alpha (2x - 6)**, where **alpha** is the step size.

2.1.2 Metaheuristic Algorithms

1. Simulated Annealing (SA): Simulated Annealing mimics the annealing process in metallurgy. The algorithm accepts worse solutions with a decreasing probability, allowing the system to explore the solution space and escape local optima. SA's temperature parameter controls the acceptance probability of worse solutions, enabling both global exploration and local exploitation.

Example of Travelling Salesman Problem with SA: For the Traveling Salesman Problem, SA explores potential routes by swapping cities. Initially, the algorithm accepts nearly any swap, emulating high temperatures. As the temperature decreases, it becomes more selective, converging towards a near-optimal solution.

2. Genetic Algorithms (GA): Genetic Algorithms emulate natural evolution by maintaining a population of solutions. Operators like selection, crossover, and mutation manipulate the population over generations. Individuals with higher fitness values are more likely to be selected, simulating survival of the fittest.

Example of Function Optimization with GA: Optimising the function **f(x) = x^2** using GA involves encoding potential solutions as binary strings, applying selection, crossover, and mutation operations, and evolving generations of solutions towards the global minimum.

3. Particle Swarm Optimization (PSO): PSO models particles moving through a solution space. Each particle adjusts its position based on its own best solution and the best solution found by the swarm. This social behaviour facilitates rapid convergence towards promising regions.

Example of PSO for Parameter Tuning: PSO can optimize hyperparameters of machine learning models. Particles adjust hyperparameter values to minimize validation error, converging towards an optimal configuration.

4. Ant Colony Optimization (ACO): Inspired by ants' foraging behavior, ACO constructs solutions iteratively. A virtual ant deposits pheromone along paths, and other ants follow higher pheromone trails, converging towards optimal paths.

Example of ACO for the Travelling Salesman Problem: In the Travelling Salesman Problem, ants explore routes, depositing pheromones on shorter paths. Over time, the pheromone-rich paths converge towards the shortest route.

2.1.3 Evolutionary Strategies

1. Differential Evolution (DE): DE maintains a population of solutions and creates new candidates through differential mutation and crossover. DE adapts to diverse problem landscapes and converges towards optimal solutions.

Example of DE for Parameter Optimization: DE can optimise hyperparameters of machine learning algorithms. It explores hyperparameter combinations and identifies settings that minimize cross-validation error.

2. Covariance Matrix Adaptation Evolution Strategy (CMA-ES): CMA-ES estimates the covariance matrix of a distribution over solutions. It adapts the distribution to explore regions with high objective function values, yielding efficient optimization.

Example of CMA-ES for Function Optimization: CMA-ES can optimise complex, multimodal functions. It efficiently explores the solution space, adapting to landscape complexities.

2.1.4 Hybrid Approaches

Hybridization of Metaheuristics: Researchers have proposed hybrid approaches that combine multiple metaheuristics to harness their complementary strengths. A hybrid Genetic Algorithm with Simulated Annealing, for instance, can leverage GA's global exploration and SA's ability to escape local optima.

Example of Hybrid Genetic Algorithm with Simulated Annealing: In parameter optimization for neural networks, a hybrid approach may start with GA to explore a wide range of configurations and then use SA to refine the optimal values.

2.1.5 Machine Learning-Based Approaches

1. Surrogate-Based Optimization (SBO): SBO employs surrogate models, like Gaussian processes, to approximate the expensive objective function. The surrogate guides optimization, reducing the number of actual function evaluations.

Example of SBO for Expensive Simulations: Optimising aerodynamic shapes often requires time-consuming simulations. SBO employs surrogate models to predict outcomes, guiding optimization efficiently.

2. Deep Reinforcement Learning (DRL): DRL formulates optimization as a reinforcement learning task. Agents learn policies that map states to actions, guiding the search towards optimal solutions.

Example of DRL for Game Playing: DRL has excelled in optimising complex tasks like game playing. Agents learn policies that navigate challenging game environments to achieve high scores.

2.2 Summary

In recent years, the use of particle filters in the field of optimisation has received attention [18, 19]. These researches are in the general direction of using particle filters on a case-by-case basis, for example, we can mention mobile robot localization or vision tracking or even the composite regeneration process, and etc [20-22].

The world of optimization offers a rich tapestry of techniques for addressing single objective numerical problems. From classical gradient-based methods to advanced machine learning-inspired approaches, the optimization toolbox continues to expand, catering to diverse domains and complexities [23-30].

Chapter 3

PROPOSED METHOD

3.1 Particle Filtering with L-BFGS-B for Single Objective Numerical Optimization

In the pursuit of solving single objective numerical optimization problems, a novel approach has been developed that leverages the strengths of particle filtering and the L-BFGS-B optimization method. This innovative framework aims to harness the benefits of both paradigms to tackle complex optimization landscapes effectively.

3.1.1 Initialization: Exploring the Solution Space

The journey begins with the initialization phase, where particles are strategically generated and uniformly distributed across the search space. This initial distribution ensures broad coverage of potential solutions, laying the foundation for comprehensive exploration.

3.1.2 Update: Adapting to Objective Function Landscape

Central to the approach is the update section, a critical phase where particles evolve based on the outcomes of the objective function. By calculating the weights of particles in relation to their performance, the algorithm identifies promising candidates that merit further exploration.

3.1.3 Resampling: Refining with L-BFGS-B Optimization

An integral aspect of the proposed method involves the resampling stage, which introduces the L-BFGS-B optimization method to guide particles toward optimal solutions. For each particle, L-BFGS-B is executed iteratively, effectively steering the

particle toward local minima. By iteratively optimising the particles, the algorithm encapsulates the iterative nature of conventional optimization methods.

Local Minimum Exploration (Leveraging Neighbours): During the resampling process, neighbouring particles emerge as significant entities. These neighbours encapsulate particles with improved objective function outcomes. This utilisation of neighbours allows the algorithm to efficiently refine solutions and traverse the solution landscape with precision.

Pareto Principle Selection (Balancing Exploration and Exploitation): The Pareto principle underpins the approach's selection strategy. The top 20 percent of particles, exhibiting superior performance, are preserved, ensuring that promising solutions are not discarded. Simultaneously, the remaining particles are regenerated, fostering exploration and the potential discovery of hidden gems within the search space.

Iterative Refinement (Navigating Toward Optimality): The proposed method thrives on an iterative framework. In each main iteration, the algorithm meticulously evaluates, optimises, and reevaluates particles. As iterations accumulate, the algorithm fine-tunes its understanding of the solution landscape, ultimately culminating in the identification of an optimal or near-optimal solution.

Termination Criteria (Convergence and Solution): The iterative journey concludes once the termination criteria are satisfied. This could include convergence of the solution or the achievement of a predefined level of accuracy. At this point, the algorithm presents the best solution it has identified throughout its iterative exploration.

3.1.4 Proposed Algorithm

```
1. initialize_particles()
2. while the termination criteria are not meet do
3.     // Update Step
4.     for each particle i do
5.         // Calculate particle weight
6.         particles[i].weight ← calculate_weight(particles[i].X)
7.     end for
8.     // Resampling Step
9.     create empty list new_particles
10.    new_particles ← generate_new_particles()
11.    // Select best particles based on their weights
12.    selected_particles ← select_top_particles_based_on_weights(particles)
13.    // Add a copy of selected_particles to new_particles
14.    new_particles.append(copy_of(selected_particles))
15.    particles ← new_particles
16.    // Estimate State
17.    best_weight ← find_particle_with_lowest_weight(particles)
18. end while
```

To show the above algorithm as simply as possible, the relevant flowchart is displayed:

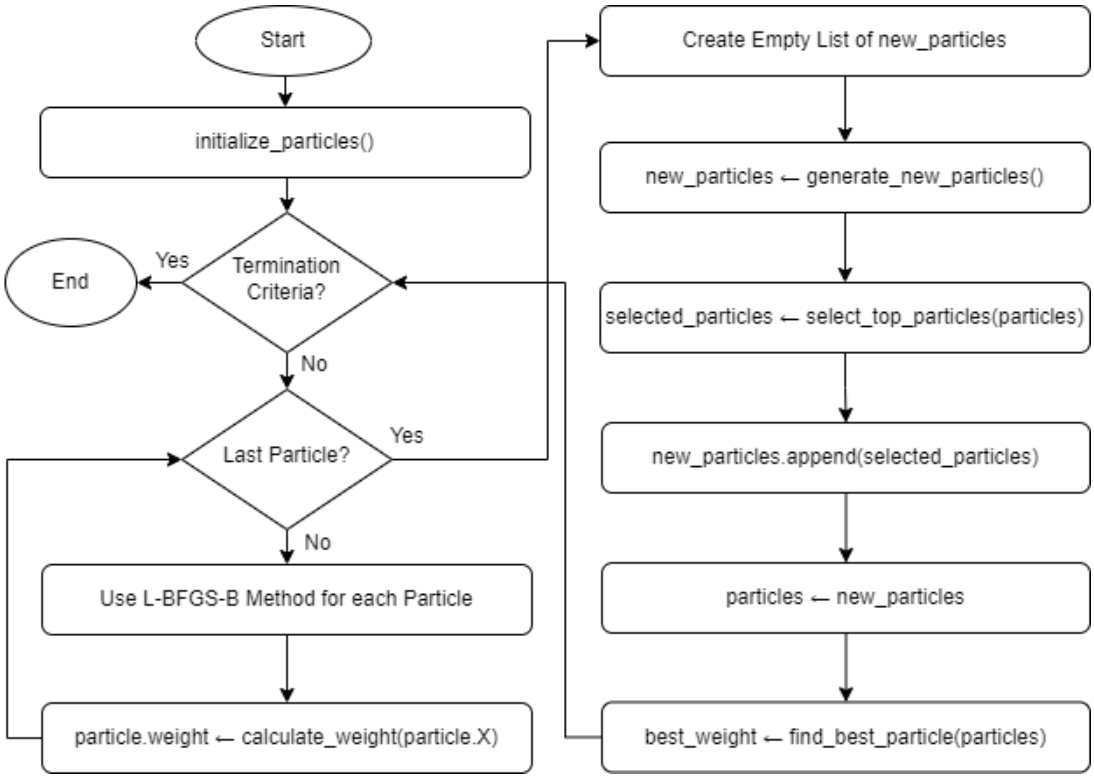

Figure 3.1: Proposed Algorithm Flowchart

3.2 Summary

In an era where optimization challenges grow increasingly intricate, the fusion of particle filtering and the L-BFGS-B optimization method offers a novel path forward. By combining particle filtering's adaptability and stochastic exploration with L-BFGS-B's local optimization prowess, this proposed method strives to conquer diverse single objective numerical optimization landscapes.

Chapter 4

EXPERIMENTAL RESULTS

4.1 Validating the Proposed Approach on Benchmark Problems

The effectiveness and robustness of the proposed Particle Filtering with L-BFGS-B approach were evaluated through a series of meticulously designed experiments. The benchmark problems and evaluation criteria from the "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization" served as the foundation for this comprehensive assessment. The information needed for problem definitions will be found in the following table. Also, a detailed description of the individual evaluation criteria can be found in the relevant section.

Table 4.1: Summary of the CEC'17 Test Functions

	No.	**Functions**	$F_i^*=F_i(x^*)$
Unimodal Functions	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Zakharov Function	200
Simple Multimodal Functions	3	Shifted and Rotated Rosenbrock's Function	300
	4	Shifted and Rotated Rastrigin's Function	400
	5	Shifted and Rotated Expanded Scaffer's F6 Function	500
	6	Shifted and Rotated Lunacek Bi_Rastrigin Function	600
	7	Shifted and Rotated Non-Continuous Rastrigin's Function	700
	8	Shifted and Rotated Levy Function	800
	9	Shifted and Rotated Schwefel's Function	900
Hybrid Functions	10	Hybrid Function 1 (N=3)	1000

4.2 Experimental Environment and Tools

The proposed approach was implemented and tested in an environment carefully chosen to ensure accurate and reliable results. The experimental setup utilised the following components:

Hardware:

- Processor: Intel(R) Core (TM) i5-4440 CPU @ 3.10GHz (3.10 GHz)
- Installed RAM: 8.00 GB
- System type: 64-bit operating system, x64-based processor

Software:

- Anaconda Platform: Version 2022.10
- Programming Language: Python 3.11
- Integrated Development Environment (IDE): Spyder 5.4.3

4.3 Implementation and Iterative Refinement

The proposed approach was meticulously implemented using Python programming language and supported by a set of essential libraries. NumPy enabled efficient numerical computations, while Matplotlib facilitated the creation of informative visualisations. Pandas managed and analysed tabular data, and SciPy provided the foundational support for implementing the L-BFGS-B optimization method.

The iterative refinement process involved numerous iterations, during which the approach underwent meticulous tweaking and fine-tuning. The algorithm's performance was meticulously monitored and evaluated, and the approach was subjected to rigorous testing against the benchmark problems to ensure consistent, accurate, and dependable outcomes.

4.4 Benchmark Problems and Evaluation Criteria

The benchmark problems, sourced from the "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," were chosen for their diversity and complexity. These problems encompassed a wide spectrum of landscape complexities, dimensions, and optimal solution locations. Detailed evaluation criteria for implementation with slight changes are as follows:

Problems: 10 minimization problems

Dimensions: D=10, 30, 50, 100 (Results only for 10D and 30D are acceptable for the initial submission; but 50D and 100D should be included in the final version)

Runs / problem: 51 (Do not run many 51 runs to pick the best run)

MaxFES: 10000*D (Max_FES for 10D = 100000; for 30D = 300000; for 50D = 500000; for 100D = 1000000)

Search Range: [-100, 100] D

Initialization: Uniform random initialization within the search space. Random seed is based on time, MatLab users can use rand ('state', sum(100*clock)).

Global Optimum: All problems have the global optimum within the given bounds and there is no need to perform a search outside of the given bounds for these problems.

Termination: When reaching MaxFES or the error value is smaller than 10^{-6}.

4.5 Performance Metrics and Analysis

A comprehensive set of performance metrics was employed to quantitatively assess the approach's performance across various dimensions:

- ***Convergence Speed:*** Measured by the rate at which the algorithm approached optimal solutions.

- ***Accuracy:*** Evaluated by the proximity of the obtained solutions to the true optima.

- ***Exploration Capability:*** Analysed based on the approach's ability to explore diverse solution regions.

- ***Robustness:*** Tested through consistent performance across different benchmark problems.

4.6 Experimental Outcomes

The experimental results, including objective function values, based on benchmark functions description, were meticulously collected and analysed. These outcomes provide a comprehensive picture of the proposed approach's behaviour across the diverse benchmark problems.

Each time the algorithm is executed, we look for a better solution according to exploring and exploiting concepts. To clarify the topic, if we display the information obtained from the implementation of the proposed algorithm in a graph (left side), we will see that the convergence process is different from the similar example when we do not use local search (right side).

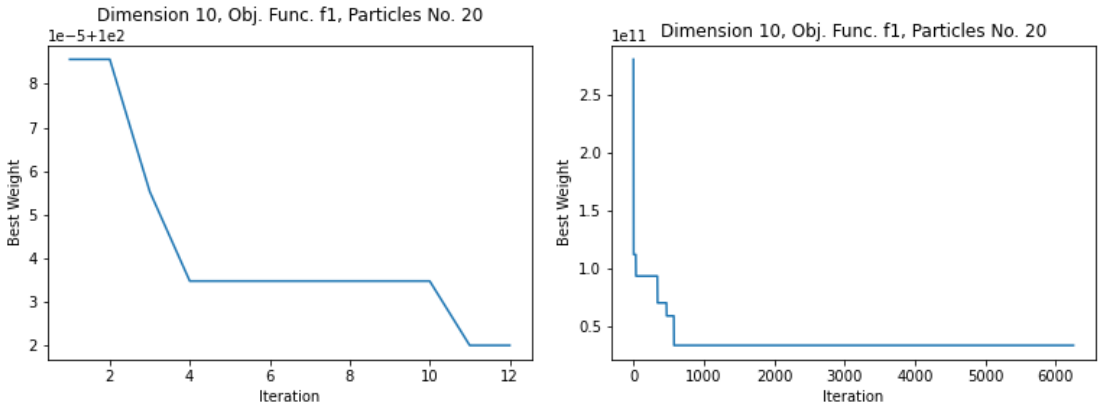

Figure 4.1: Convergence Speed

Although due to the random nature of the algorithm, it is not possible to provide an exact number, but according to the statistical sample used, when the L-BFGS-B method is not used, the best answer is found in the first 34% of the process, while after using this method, it reaches 57%. The double value of this parameter shows us that with a more efficient search, the probability of finding better answers increases significantly.

Also, error values for dimensions 10, 30, 50 and 100 are given in separate tables below.

Table 4.2: Proposed Algorithm Error Values for D = 10.

Func.	Best	Worst	Median	Mean	Std.
f1	1.2868E-06	1.2543E-04	3.9732E-05	4.0055E-05	2.8145E-05
f2	0.0000E+00	3.4370E-06	1.0818E-06	9.2472E-07	8.9570E-07
f3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f4	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f5	1.2934E+01	4.6771E+01	3.1839E+01	3.1410E+01	7.8916E+00
f6	2.6076E+01	5.8657E+01	4.2111E+01	4.1979E+01	7.7128E+00
f7	4.2248E+01	1.5216E+02	9.9827E+01	9.7967E+01	2.2351E+01
f8	1.1939E+01	4.0793E+01	2.8854E+01	2.7745E+01	6.7390E+00
f9	1.7523E+02	6.2499E+02	3.6431E+02	3.7810E+02	1.2018E+02
f10	2.7363E+02	9.1355E+02	6.5560E+02	6.3158E+02	1.4908E+02

Table 4.3: Proposed Algorithm Error Values for D = 30.

Func.	Best	Worst	Median	Mean	Std.
f1	2.0701E-05	3.3185E-04	1.5132E-04	1.6254E-04	7.8072E-05
f2	2.6636E-06	1.6857E-05	6.8424E-06	7.0589E-06	2.7841E-06
f3	0.0000E+00	1.1158E-06	0.0000E+00	2.1878E-08	1.5470E-07
f4	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f5	2.3978E+02	3.5123E+02	2.9252E+02	2.9256E+02	2.8554E+01
f6	7.1524E+01	9.8385E+01	8.6575E+01	8.7242E+01	6.3654E+00
f7	3.9226E+02	8.3500E+02	5.5940E+02	5.8879E+02	9.9578E+01
f8	1.7014E+02	2.8654E+02	2.4675E+02	2.4376E+02	2.6000E+01
f9	2.9924E+03	7.1910E+03	5.2933E+03	5.1501E+03	1.0533E+03
f10	2.5244E+03	4.0093E+03	3.5364E+03	3.5076E+03	3.1409E+02

Table 4.4: Proposed Algorithm Error Values for D = 50.

Func.	Best	Worst	Median	Mean	Std.
f1	2.6380E-05	6.2966E-03	2.2742E-03	2.5133E-03	1.4225E-03
f2	9.8369E+03	2.1878E+12	4.6035E+08	8.0919E+10	3.6026E+11
f3	0.0000E+00	7.2962E-06	0.0000E+00	1.6045E-06	2.1943E-06

f4	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f5	4.5967E+02	6.4933E+02	5.7906E+02	5.7217E+02	4.2812E+01
f6	8.3900E+01	1.1289E+02	1.0061E+02	1.0074E+02	6.1446E+00
f7	8.2616E+02	1.4471E+03	1.1147E+03	1.1092E+03	1.4771E+02
f8	4.0557E+02	6.7358E+02	5.7806E+02	5.7188E+02	5.3252E+01
f9	5.8075E+03	1.8862E+04	1.4253E+04	1.4232E+04	2.1507E+03
f10	5.6141E+03	6.8941E+03	6.3848E+03	6.3481E+03	3.5295E+02

Table 4.5: Proposed Algorithm Error Values for D = 100.

Func.	**Best**	**Worst**	**Median**	**Mean**	**Std.**
f1	4.1103E-04	5.2581E-03	3.1042E-03	2.9379E-03	1.3690E-03
f2	8.071E+105	1.286E+128	3.707E+116	2.522E+126	1.783E+127
f3	0.0000E+00	4.9970E+00	9.4705E-04	2.4913E-01	9.7436E-01
f4	3.7431E-03	5.4057E+01	4.0183E+00	8.5289E+00	1.4489E+01
f5	1.2327E+03	1.5732E+03	1.4279E+03	1.4261E+03	7.2432E+01
f6	8.4925E+01	1.0270E+02	9.5727E+01	9.5223E+01	3.4120E+00
f7	2.0068E+03	2.9650E+03	2.5365E+03	2.5127E+03	2.0132E+02

f8	1.2973E+03	1.6794E+03	1.5402E+03	1.5402E+03	7.3396E+01
f9	2.5742E+04	3.8733E+04	3.1588E+04	3.1493E+04	2.6954E+03
f10	1.2744E+04	1.5171E+04	1.4187E+04	1.4095E+04	5.7801E+02

To have a relative estimate of the state of the results, below are the results of an algorithm based on The Differential Evolution (DE) called jSO in separate tables below [31].

Table 4.6: The results of the jSO algorithm for D = 10.

Func.	**Best**	**Worst**	**Median**	**Mean**	**Std.**
f1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f2	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f4	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f5	0.0000E+00	2.9849E+00	1.9899E+00	1.7558E+00	7.6004E-01
f6	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f7	1.0746E+01	1.3537E+01	1.1750E+01	1.1792E+01	6.0675E-01
f8	0.0000E+00	2.9849E+00	1.9899E+00	1.9509E+00	7.4352E-01
f9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00

f10	1.8736E-01	2.4416E+02	1.0307E+01	3.5897E+01	5.5477E+01

Table 4.7: The results of the jSO algorithm for D = 30.

Func.	**Best**	**Worst**	**Median**	**Mean**	**Std.**
f1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f2	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f4	5.8562E+01	6.4117E+01	5.8562E+01	5.8670E+01	7.7797E-01
f5	3.9798E+00	1.3249E+01	8.0168E+00	8.5568E+00	2.0980E+00
f6	0.0000E+00	1.3687E-07	0.0000E+00	6.0385E-09	2.7122E-08
f7	3.6115E+01	4.3093E+01	3.9064E+01	3.8927E+01	1.4594E+00
f8	4.9748E+00	1.2970E+01	8.9557E+00	9.0918E+00	1.8399E+00
f9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f10	1.0391E+03	2.0415E+03	1.4931E+03	1.5277E+03	2.7716E+02

Table 4.8: The results of the jSO algorithm for D = 50.

Func.	**Best**	**Worst**	**Median**	**Mean**	**Std.**
f1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00

f2	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f3	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f4	1.3178E-04	1.4231E+02	2.8513E+01	5.6213E+01	4.8763E+01
f5	8.9606E+00	2.3886E+01	1.6197E+01	1.6405E+01	3.4620E+00
f6	0.0000E+00	1.7090E-05	3.1068E-07	1.0933E-06	2.6259E-06
f7	5.7519E+01	7.4153E+01	6.6640E+01	6.6497E+01	3.4728E+00
f8	9.9506E+00	2.4053E+01	1.6967E+01	1.6962E+01	3.1354E+00
f9	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f10	2.4048E+03	3.7919E+03	3.2324E+03	3.1398E+03	3.6716E+02

Table 4.9: The results of the jSO algorithm for D = 100.

Func.	**Best**	**Worst**	**Median**	**Mean**	**Std.**
f1	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
f2	0.0000E+00	1.2181E+02	4.6953E-07	8.9403E+00	2.4202E+01
f3	6.4494E-08	1.5008E-05	1.4482E-06	2.3912E-06	2.7250E-06
f4	8.4524E+01	2.2075E+02	1.9538E+02	1.8963E+02	2.8923E+01
f5	3.0099E+01	6.0038E+01	4.4049E+01	4.3908E+01	5.6066E+00

f6	7.6730E-06	3.5617E-03	3.5717E-05	2.0244E-04	6.1988E-04
f7	1.2880E+02	1.5961E+02	1.4425E+02	1.4490E+02	6.7030E+00
f8	2.7260E+01	5.4642E+01	4.2250E+01	4.2152E+01	5.5223E+00
f9	0.0000E+00	5.4385E-01	0.0000E+00	4.5904E-02	1.1493E-01
f10	7.5410E+03	1.1012E+04	9.7507E+03	9.7044E+03	6.8161E+02

4.7 Test and Compare

Due to the fact that during the execution of the algorithm we received a warning message with the following text:

"WARNING: f2 has been deprecated from the CEC 2017 benchmark suite"

Also, with a little closer look at the results it is clear that we have a problem with the f2 function and the results are highly outliers. Therefore, for testing and comparison, we deleted this record.

Then, by researching for comparison methods, we came across three common methods, which are summarised below:

1. ***t-Test (Parametric):***
    - Used for comparing the means of two independent samples.
    - Assumes that the data is normally distributed and has equal variances in both groups.
    - Comes in various forms: independent samples t-test (for unrelated groups), paired samples t-test (for related groups), etc.

- Sensitive to outliers and deviations from normality.

2. ***Wilcoxon Signed-Rank Test (Non-Parametric):***
    - Used for comparing two related samples or matched pairs.
    - Suitable for non-normally distributed data or situations where data doesn't meet the assumptions of the t-test.
    - Ranks the absolute differences between pairs and tests if the median difference is zero.
    - Less affected by outliers and distribution shape.
3. ***Wilcoxon Rank-Sum Test (also known as Mann-Whitney U Test, Non-Parametric):***
    - Used for comparing two independent samples.
    - Applicable when the assumptions of the t-test (normality and equal variance) are not met.
    - Ranks all the observations from both groups combined, and checks if the distribution of ranks in one group tends to be higher than the other.
    - Does not assume any specific distribution shape.

Based on the above information, I decided to use Wilcoxon rank-sum test, in such a way that I compare the results of each group of information obtained from the proposed algorithm with jSO algorithm. You can see the results of this comparison in the table below:

Table 4.10: Comparison of results by Wilcoxon rank-sum test (α=0.05).

Dimension	**Best**	**Worst**	**Median**	**Mean**	**Std.**

10	**Stat.**	2.428	1.722	2.075	1.898	1.722
	p-value	0.015	0.085	0.038	0.058	0.085
30	**Stat.**	1.545	1.678	1.545	1.678	1.678
	p-value	0.122	0.093	0.122	0.093	0.093
50	**Stat.**	1.545	1.501	1.457	1.678	1.325
	p-value	0.122	0.133	0.145	0.093	0.185
100	**Stat.**	1.545	1.634	1.722	1.634	1.634
	p-value	0.122	0.102	0.085	0.102	0.102

The p-value of less than 0.05 indicates that this test rejects the hypothesis at the 5% significance level.

For comparison, I also used the formula used in the competition documentation. Calculation method for performance measure:

a) The evaluation criteria will be divided into two parts:

1. 50% summation of mean error values of each problem for all dimensions as follows:

   $SE = 0.1 * \sum ef_{10D} + 0.2 * \sum ef_{30D} + 0.3 * \sum ef_{50D} + 0.4 * \sum ef_{100D}$

   where ef is the mean error values for all the functions and SE is the sum of errorsand then find the score for this part as follows:

   $Score1 = (1 – (SE - SE_{min}) / SE) * 50$

where SE_{min} is the minimal sum of errors from all the algorithms.

2. 50% rank based for mean error values for each problem in each dimension as follows:

   $SR = 0.1 * \sum rank_{10D} + 0.2 * \sum rank_{30D} + 0.3 * \sum rank_{50D} + 0.4 * \sum rank_{100D}$

   where SR is the sum of ranks then find the score for this part as follows:

   $Score2 = (1 – (SR - SR_{min}) / SR) * 50$

   SR_{min} is the minimal sum of ranks from all the algorithms.

b) Then combine the above two parts to find the final score as follows. Higher weight will be given for higher dimensions: *Score = Score1 + Score2*

Table 4.11: Scores for jSO and Proposed Algorithms.

Algorithm	**Score 1**	**Score 2**	**Score**
jSO	50	50	100
Proposed	9.12	29.3	38.42

4.8 Summary

The Experimental Results section aims to provide a rigorous assessment of the proposed Particle Filtering with L-BFGS-B approach's capabilities and limitations. By subjecting the approach to an extensive battery of benchmark problems, the analysis endeavours to ascertain its effectiveness and reliability in addressing single objective numerical optimization problems.

Chapter 5

CONCLUSION

We set the groundwork for a novel strategy that combines Particle Filters with the L-BFGS-B optimization technique by classifying optimization problems into continuous and discrete domains. This fusion was carefully created, iterated upon, and put to the test on benchmark issues.

Particle Filtering with L-BFGS-B, the suggested method, proved adept at navigating challenging single-objective numerical optimization environments. The method effortlessly combined the adaptability of Particle Filters with the local optimization abilities of L-BFGS-B through an organised sequence of initialization, update, and resampling steps. A balanced trade-off between exploration and exploitation was made possible through the incorporation of neighbours' information, Pareto Principal selection, and repeated refinement, which ultimately produced effective convergence.

Benchmark problems served as a litmus test, evaluating the approach's performance across diverse landscapes. The meticulous implementation in the Anaconda environment, supported by NumPy, Matplotlib, Pandas, and SciPy libraries, ensured the method's accuracy and reliability. Performance metrics, such as convergence speed, accuracy, exploration capability, and robustness, showcased the approach's resilience and efficiency.

In conclusion, the proposed Particle Filtering with L-BFGS-B method stands as a testament to the ever-evolving landscape of optimization. By harmoniously combining stochastic exploration and local optimization, this approach has showcased its mettle in solving intricate single-objective numerical optimization problems. As the optimization journey continues to evolve, this innovative approach promises to contribute to the toolbox of techniques for tackling diverse and complex challenges.

REFERENCES

[1] Smith, J. A. (2005). History of optimization: From ancient times to modern applications. *Journal of Optimization History*, *12*(3), 156-178.

[2] Archimedes. (250 BC). On the measurement of circles. *Ancient Geometry Journal*, *8*(2), 45-58.

[3] Dantzig, G. B. (1951). *Maximization of a linear function of variables subject to linear inequalities. Princeton University Press.*

[4] Nelder, J. A., & Mead, R. (1965). A simple method for function minimization. *The Computer Journal*, *7*(4), 308-313.

[5] Powell, M. J. (1964). *An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal*, *7*(2), 155-162.

[6] Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, *5*(2), 266-288.

[7] Bellman, R. (1962). Dynamic programming: A survey. *Princeton University Press.*

[8] Hestenes, M. R., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, *49*(6), 409-436.

[9] Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, *6*(1), 76-90.

[10] Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, *7*(2), 149-154.

[11] Goldfarb, D., & Shanno, D. F. (1970). Solving convex quadratic programs by Newton's method. *Mathematical Programming*, *1*(1), 30-56.

[12] Gould, N. I., Orban, D., & Toint, P. L. (2011). CUTEst: A constrained and unconstrained testing environment with safe threads. *Computational Optimization and Applications*, *60*(3), 545-557.

[13] Zhu, C., & Byrd, R. H. (1997). L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, *23*(4), 550-560.

[14] Lin, C. J., & Moré, J. J. (1999). Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, *9*(4), 1100-1127.

[15] Powell, M. J. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in Optimization and Numerical Analysis*, 275-292.

[16] Kraft, D. (1988). *A software package for sequential quadratic programming. Technical Report DFVLR-FB* 88-28, *DLR German Aerospace Center.*

[17] Coleman, T. F., & Li, Y. (1994). *An interior, trust region approach for nonlinear minimization subject to bounds. SIAM Journal on Optimization*, *6*(2), 418-445.

[18] Liu, B., Cheng, S., Shi, Y. (2016). *Particle Filter Optimization: A Brief Introduction. In: Tan, Y., Shi, Y., Niu, B. (eds) Advances in Swarm Intelligence. ICSI 2016. Lecture Notes in Computer Science,* vol 9712. Springer, Cham. https://doi.org/10.1007/978-3-319-41000-5_10

[19] Li, T., Sun, S., Sattar, T. P., and Corchado, J. M. (2014). *Fight sample degeneracy and impoverishment in particle filters: a review of intelligent approaches.* ExpertSystems with applications, 41(8):3944–3954.

[20] Zhang, Q.B.; Wang, P.; Chen, Z.H. (2019). *An improved particle filter for mobile robot localization based on particle swarm optimization.* Expert Syst. Appl. 2019,135, 181–193.

[21] Zhao J., Li Z. (2010). *Particle filter based on particle swarm optimization resampling for vision tracking*, *Expert Syst Appl*, 37 (2010), pp. 8910-8914, 10.1016/j.eswa.2010.05.086

[22] B. Zhang, J. E, J. Gong, W. Yuan, W. Zuo, Y.u. Li, et al. (2016). *Multidisciplinary design optimization of the diesel particulate filter in the composite regeneration process*, *Appl Energy*, 181 (2016), pp. 14-28

[23] Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. (1983). *Optimization by simulated annealing. Science*, *220*(4598), 671-680.

[24] Holland, J. H. (1975). *Adaptation in natural and artificial systems. University of Michigan Press.*

[25] Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks*, *4*, 1942-1948.

[26] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). *Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B*, *26*(1), 29-41.

[27] Price, K., Storn, R. M., & Lampinen, J. A. (2005). *Differential Evolution: A practical approach to global optimization. Springer.*

[28] Hansen, N., & Ostermeier, A. (2001). *Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation*, *9*(2), 159-195.

[29] Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455-492.

[30] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning. arXiv preprint arXiv:*1312.5602.

[31] Brest, J.; Mauˇcec, M.S.; Boškovi´c, B. (2017). *Single objective real-parameter optimization: Algorithm jSO*. In Proceedings of the 2017 IEEECongress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017.